WHAT IS CLAIMED IS:

1. A method of scheduling packets from multiple queues onto a common output packet stream, the method comprising:

maintaining a programmable interleaving table having multiple entries, each entry identifying one of the queues with an epoch value;

maintaining a pointer to a current table entry in the interleaving table;

for each epoch in a sequence of epochs, scheduling packets onto the output packet stream from the queue identified with the current table entry, and charging that entry for use of the epoch; and

moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry.

2. The method of claim 1, wherein the number of entries in the programmable interleaving table exceeds the number of queues, such that at least some queues are identified more than once in the table, each entry containing a field explicitly identifying the queue associated with that entry.

3. The method of claim 2, wherein the epoch value for each entry is a common value shared by all entries.

4. The method of claim 2, wherein the epoch value for each entry is based on the queue identified with the entry.

5. The method of claim 2, wherein the epoch value for each entry is a programmable value occupying a field of that entry.

6. The method of claim 2, wherein each entry comprises an epoch counter field, and wherein charging an entry for use of an epoch comprises changing the value of the epoch counter field by one.

7. The method of claim 6, wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer after the value in the epoch counter has been changed a number of times corresponding to the current table entry's epoch value.

8. The method of claim 6, wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer after each epoch to the next table entry that has not yet changed the value of its epoch counter field a number of times corresponding to that entry's epoch value.

9. The method of claim 8, wherein when all table entries have been charged with use of a number of epochs corresponding to their respective epoch values, re-initializing each epoch counter field and setting the pointer to point to the head of the table.

10. The method of claim 2, wherein when the pointer is moved to a next table entry an epoch register is initialized, wherein charging an entry for use of an epoch comprises changing the value of the epoch register by one, and wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer after the value in the epoch register has

been changed a number of times corresponding to the current table entry's epoch value.

11. The method of claim 2, wherein when the queue identified with the current table entry is empty and the epoch has not yet ended, moving the pointer to the next table entry that is not empty, and scheduling packets onto the output packet stream from the queue identified with that next table entry for the remainder of the epoch.

12. The method of claim 11, wherein when the pointer is moved during an epoch, only the first table entry of the epoch is charged for use of the epoch.

13. The method of claim 1, wherein the common output packet stream serves multiples sets of queues, each set of queues having its own programmable table, and wherein maintaining a pointer to a current table entry comprises maintaining a separate pointer, for each set of queues, into that set's programmable table, the method further comprising:

selecting, for each epoch in the sequence of epochs, one of the sets of queues for packet scheduling during that epoch; and

selecting the pointer corresponding to the selected set of queues for use during that epoch.

14. The method of claim 1, wherein the number of entries in the programmable table equals the number of queues, wherein each entry comprises an epoch value field and an epoch counter field, wherein charging an entry for use of an epoch comprises changing the value of that field's epoch counter by one, and wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer, after each epoch, to the next table entry that

has not yet changed the value of its epoch counter field a number of times corresponding to that entry's epoch value.

15. An integrated circuit having a scheduler comprising:

a programmable memory capable of configuration as an interleaving table having multiple entries, each entry capable of identifying a queue with an epoch value;

a pointer register capable of identifying a current entry in the interleaving table; and

a queue sequencer to supply, for each epoch, a queue identifier based on the current entry in the interleaving table, to charge the current entry for use of the epoch, and to step the pointer register to a next table entry when the current entry has been charged with use of a number of epochs set aside for that entry.

16. The circuit of claim 15, wherein the pointer register comprises multiple pointer entries, each pointer entry capable of identifying a current entry in the interleaving table corresponding to that entry.

17. The circuit of claim 16, wherein the programmable memory is capable of configuration as multiple interleaving table segments, each pointer entry capable of identifying table entries within a corresponding one of the table segments.

18. The circuit of claim 15, wherein the queue sequencer has access to a queue status register that indicates whether each queue identified in the interleaving table is empty or not, the queue sequencer capable of using the queue status register contents to step the pointer register to a next table entry within an epoch, when the queue status of the queue identified with the current entry indicates that that queue is empty.

19. An epoch-based scheduler comprising: 370/229 ? 371 ? 235

means for storing a sequence of interleaved access to multiple queues;

means for maintaining a current position in the sequence; and

means for stepping the maintained current position to a next location in the stored sequence after the queue associated with the current position has been used for a number of epochs set aside for the current position.

20. A packet routing device comprising: (370/355) 389, 451)

a switching fabric having a plurality of input ports and a plurality of output ports, the switching fabric capable of reconfiguration on an epoch-by-epoch basis to transmit packets according to an input port-to-output port mapping specified for each epoch;

a plurality of input packet buffers, each connected to a corresponding one of the switching fabric input ports to supply packets to that input port;

associated with each input packet buffer, a queue memory capable of supporting, for each of the switching fabric output ports, an output-port-specific set of packet queues;

associated with each input packet buffer, a memory capable of configuration as a set of interleaving tables, with at least one interleaving table per output port, each interleaving table entry identified with one of the packet queues from the set of queues specified for the output port corresponding to that interleaving table; and

associated with each input packet buffer, a queue sequencer to select, for each epoch, at least one queue, from the set corresponding to the output port mapped to that input packet buffer during that epoch, to supply packets to the input packet buffer during that epoch, the queue sequencer selecting the queue identified with a current entry in the interleaving table associated with that input packet buffer and the mapped output port.

21. The device of claim 20, wherein the interleaving table associated with each input packet buffer is capable of further configuration to include a set of multicast interleaving tables, each entry in a multicast interleaving table identified with a particular multicast group queue.

22. An apparatus comprising a computer-readable medium containing computer instructions that, when executed, cause a processor or multiple communicating processors to perform a method for scheduling packets from multiple queues onto a common output packet stream, the method comprising:

maintaining a programmable interleaving table having multiple entries, each entry identifying one of the queues with an epoch value;

maintaining a pointer to a current table entry in the interleaving table;

for each epoch in a sequence of epochs, scheduling packets onto the output packet stream from the queue identified with the current table entry, and charging that entry for use of the epoch; and

moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry.

23. The apparatus of claim 22, wherein the number of entries in the programmable interleaving table exceeds the number of queues, such that at least some queues are identified more than once in the table, each entry containing a field explicitly identifying the queue associated with that entry.

24. The apparatus of claim 23, wherein the epoch value for each entry is a programmable

value occupying a field of that entry.

25. The apparatus of claim 23, wherein each entry comprises an epoch counter field, and wherein charging an entry for use of an epoch comprises changing the value of the epoch counter field by one.

26. The apparatus of claim 25, wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer after the value in the epoch counter has been changed a number of times corresponding to the current table entry's epoch value.

27. The apparatus of claim 25, wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer after each epoch to the next table entry that has not yet changed the value of its epoch counter field a number of times corresponding to that entry's epoch value.

28. The apparatus of claim 27, wherein when all table entries have been charged with use of a number of epochs corresponding to their respective epoch values, re-initializing each epoch counter field and setting the pointer to point to the head of the table.

29. The apparatus of claim 23, wherein when the queue identified with the current table entry is empty and the epoch has not yet ended, moving the pointer to the next table entry that is not empty, and scheduling packets onto the output packet stream from the queue identified with that next table entry for the remainder of the epoch.

30. The apparatus of claim 22, wherein the common output packet stream serves multiples sets of queues, each set of queues having its own programmable table, and wherein maintaining a pointer to a current table entry comprises maintaining a separate pointer, for each set of queues, into that set's programmable table, the method further comprising:

selecting, for each epoch in the sequence of epochs, one of the sets of queues for packet scheduling during that epoch; and

selecting the pointer corresponding to the selected set of queues for use during that epoch.

31. The apparatus of claim 22, wherein the number of entries in the programmable table equals the number of queues, wherein each entry comprises an epoch value field and an epoch counter field, wherein charging an entry for use of an epoch comprises changing the value of that field's epoch counter by one, and wherein moving the pointer to a next table entry when the current table entry has been charged with use of a number of epochs set aside for that entry comprises moving the pointer, after each epoch, to the next table entry that has not yet changed the value of its epoch counter field a number of times corresponding to that entry's epoch value.